Compiler Support for Message Passing Systems

Darpa Contract No. MDA972-97-C-0003

Quarterly R&D Status Report
September 27, 1997 – December 31, 1997

Scientific Computing Associates, Inc.
New Haven, CT

Jens Nielsen, Principal Investigator

January 31, 1998

1. Progress During Reporting Period

During this period, we continued to develop the run-time system for our prototype enhanced MPI tools. The primary focus was on development of a more complete implementation of the error checking facility for standard MPI *send* and *receive* operations, including the ability to handle MPI derived data types. This effort conforms to the approach laid out in our proposal for the core project; it addresses objectives *e.4.1*, *e.4.2*, *e.4.3,* and *e.4.5* in our work plan.

In addition to the technical work, we took part in the following activities in relation to the work in this project:

1. We submitted a proposal to Lincoln Laboratory under which Scientific Computing Associates, Inc. will investigate the use of networks of workstations for the MPI version of Lincoln Laboratory's signal processing library. We have received a purchase order and anticipate beginning this effort in February, 1998. While the work is not directly related to this SBIR project (since none of the technology from this project will be used), we do plan to examine the signal processing library as part of our assessment of "real" MPI codes.
2. We submitted a request to Darpa for funding of the options associated with this SBIR project. According to the contract, the options are available for funding as of May 15, 1998, although we anticipate that they may be delayed due to funds availability.

1.1 Enhanced MPI Prototype Development

1.1.1 Run-Time Error Checking

The main focus of our effort in this period has been extension of the run-time error checking functionality to handle MPI derived data types. As noted in our previous report, run-time error checking is primarily intended to detect situations in which the sender and receiver disagree as to the interpretation of the contents of a message payload. The simplest sort of example might be

one in which the sender packs an integer followed by a float into a message, but the receiver unpacks a float followed by an integer. With the work done in this quarter, our prototype system is now able to detect errors involving MPI derived data types, either alone or in combination with standard MPI basic data types. For example, the system can detect the error in the following erroneous program:

```
#include <stdio.h>
#include "mpi.h"

main(argc, argv)
     int    argc;
     char   *argv[];
{
  int       my_rank;
  float     f[30];
  int       g[10];
  char      c;

  MPI_Datatype newt1, newt2;
  MPI_Status status;
  int block_lengths[3];
  MPI_Aint  displacements[3];
  MPI_Datatype    typelist[3];
  MPI_Aint  base_address;
  MPI_Aint  address;

  MPI_Init(&argc, &argv);
  MPI_Comm_rank(MPI_COMM_WORLD, &my_rank);

  block_lengths[0] = 20;
  block_lengths[1] = 10;
  block_lengths[2] = 1;

  MPI_Address(base_address, &base_address);
  MPI_Address(f, &address);
  displacements[0] = address - base_address;
  MPI_Address(g, &address);
  displacements[1] = address - base_address;
  MPI_Address(&c, &address);
  displacements[2] = address - base_address;

  typelist[0] = MPI_FLOAT;
  typelist[1] = MPI_INT;
  typelist[2] = MPI_CHAR;

  MPI_Type_struct(3, block_lengths, displacements, typelist, &newt1);
  MPI_Type_commit(&newt1);

  MPI_Send((void *)base_address, 1, newt1, my_rank, 3, MPI_COMM_WORLD);

  block_lengths[0] = 25;
  block_lengths[1] = 5;
  block_lengths[2] = 1;

  MPI_Type_struct(3, block_lengths, displacements, typelist, &newt2);
  MPI_Type_commit(&newt2);

  MPI_Recv((void *)base_address, 1, newt2, my_rank, 3,
           MPI_COMM_WORLD,&status);

  MPI_Finalize();
  return 0;
}
```

When run, the program generates the following output:

```
Packing: (MPI_FLOAT:20)(MPI_INT:10)(MPI_CHAR:1)
Packing type MPI_FLOAT(2), size 20
Packing type MPI_INT(1), size 10
Packing type MPI_CHAR(7), size 1
Send 1
Recv 1
[derived.c:42];[derived.c:51] types out of sync.
```

Note that most of these lines of output are debugging outputs that we use to make certain that our system is operating correctly. In actual use, the user would see only the last line, which indicates the file line numbers of the mismatched operations (lines 42 and 51 in this case). We are working on more descriptive messages to indicate the specific nature of the mismatch.

At the present time, we only provide the error checking facility for invocations of `MPI_Send()` and `MPI_Recv()`, but we do not anticipate any difficulty in extending it to MPI's other variants of these operations.

2. Planned Activities and Milestones

We will continue the development of our MPI prototype software, focusing particularly on improvements in the runtime implementation and on better integration with standard MPI. The specific objectives in the next quarter are:

1. *Investigation of ways to improve run-time performance:* We plan to study the MPICH implementation of MPI in order to identify opportunities where our language-based approach can lead to run-time improvements. Specifically, we will try to determine whether the run-time system can use information about message tags and/or variable types to allow it to exploit "customized" message handlers that are significantly more efficient than the generic message handlers used by default. The idea would be to generate the handlers automatically at compile-time or link-time, much as is done for our Linda systems.
2. *Examination of MPI applications:* We plan to obtain and begin working with a number of "real" MPI codes, with two goals in mind:
   a. We would like to better understand which portions of the full MPI system are really used in practice, so that we can refine the development plan for our prototype system, particularly with respect to the assessment of opportunities for enhancing run-time performance as noted above.
   b. We would like to assess the value of the work we have done so far by testing it on more complex applications than the simple test cases we have used so far.

3. Administrative Information

No significant problems have arisen in this period, and there are no areas of concern. The core portion of the project is on schedule with respect to technical development, and the cost is consistent with the expenditure plan. There were no changes in key personnel during this period, and there were no purchases of major equipment in this period.

As noted above, we have requested that Darpa fund the two options associated with this SBIR project as of their originally requested start dates of May 15, 1998. However, it is our understanding that there may have been some confusion as to the start dates for the option

projects, with the result that funds may not be available on the original schedule in this fiscal year. Should Darpa so wish, it would be acceptable to delay the start of the option projects until funds are available at the beginning of the next fiscal year. Such a delay will not interfere with our ability to retain the services of key personnel required to complete the option projects, since those personnel will remain active on the Core Project until October 1998 in any event.

Personnel Hours		
	Planned	Actual
Current Period	424	424
Contract Since Inception	2482	2482

Expenditures in current period: $ 49,583 (inclusive of fee)

Expenditures since inception: $ 277,301 (inclusive of fee)

Total funds committed: $ 374,733

Estimated funds for completion of Core Project: $ 97,432

Approximate quarterly breakout of anticipated payments from DARPA:

$ 45,000 per calendar quarter through 2Q1998;
$ 60,000 in 3Q1998;
$ 16,211 in 4Q1998.

Estimated date of completion of Core Project: October 15, 1998